Computer Science 271
Project `0000`
Due Wednesday, January 25

To loosen up those rusty programming fingers, you will design and implement a `Set` ADT as a template class in C++. A set is an *unordered* collection of *unique* elements. For example,

$$\{1, 8, 3\}$$

is a proper set, but

$$\{1, 8, 3, 1\}$$

is not.

Your ADT should support the following operations:

- `s.insert(x)` inserts an element `x` into the set `s`
- `s.remove(x)` removes an element `x` from the set `s`
- `s.cardinality()` returns the cardinality of (i.e., number of elements in) the set `s`
- `s.empty()` indicates whether the set `s` is the empty set ($S = \emptyset$)
- `s.contains(x)` indicates whether an element `x` is contained in the set `s` ($x \in S$)
- `s == t` indicates whether the set `s` contains the same elements as the set `t` ($S = T$)
- `s <= t` indicates whether the set `s` is a subset of the set `t` ($S \subseteq T$)
- `s + t` returns the union of the set `s` and the set `t` ($S \cup T$)
- `s & t` returns the intersection of the set `s` and the set `t` ($S \cap T$)
- `s - t` returns the difference of the set `s` and the set `t` ($S \setminus T$)

Part 1: Implement a Set ADT

Implement your `Set` template class with a singly linked list. After you implement each `Set` method, write a unit test method in a separate unit test file that *thoroughly* tests that method. Test each method with the new tests and all the previous tests as well (regression testing).

In addition to the methods and operators above, your class should include a copy constructor and destructor, and overload the assignment operator and stream insertion operator (`<<`).

There are some files to get you started in the directory `/share/havill/271/proj0/` on the computers in Olin 219.

Part 2: Use your Set ADT

Look at the file `pres.txt`,[1] which contains a table of data about the 44 U.S. Presidents. Reading from left to right, the columns indicate the president's name, political party ((F) = Federalist;

[1] Adapted from `http://www.infoplease.com/ipa/A0194030.html`

(DR) = Democratic-Republican; (D) = Democratic; (W) = Whig; (R) = Republican; (U) = Union), home state, religion, and age when he took office. To make the file easier to read, each field is separated by a tab and no field contains any spaces.

Using this file, write a program that enables one to write expressions that use your `Set` class to answer questions like:

- Which presidents from Ohio were Methodist?
- Which presidents from Virginia were Episcopalians and in the Whig party?
- Which presidents were either Whig or Democratic-Republicans?
- Which presidents were in their forties when they took office?

For example,

```
cout << (episcopalian & va & whig);
```

should answer the second question above by printing

```
{W.H.Harrison, Tyler, Taylor}
```

To limit the number of sets you need to create, your program only needs to be able to handle questions that involve:

- the states VA, NY, MA, OH, and "Other"
- the religions Episcopalian, Presbyterian, Methodist, and "Other"
- decades of ages (forties, fifties, and sixties)
- the political parties Whig, Democrat, Republican, and "Other"

Your programs *must* compile with the accompanying `Makefile`.

Your final submission should consist of 5 files — `set.h`, `set.cpp`, `test_set.cpp`, and `presidents.cpp` — plus a single PDF containing the text of these 4 files. Submit these files via Notebowl by the due date. To create the PDF,

1. Change into the directory containing your code for the project.
2. In the terminal, type

   ```
   enscript -p- set.h set.cpp test_set.cpp presidents.cpp | ps2pdf - > proj0_yourname.pdf
   ```

   With this command, you are combining 4 source files for this project into one pdf named `proj0_yourname.pdf`. Obviously, substitute your own name! For future projects, do the same thing; just substitute the names of the source files and the project number in the destination PDF file name.